

Titlu proiect - The Lost Treasure

Autor - Epure Andrei-Ioan

Grupa 1209A

Povestea jocului:

Povestea se axeaza pe un tanar numit John ,acesta este complet captivat de o legenda pe care obisnuia sa o auda de la bunicul sau cand era mic.Aceasta legenda povesteste cum ca pe o insula ascunsa se afla o comoara imensa,dar aceasta este pazita de diferite creaturi infriosatoare .John odata devenit adult decide sa devina un pirat si sa navigheze oceanele in cautarea comorii.Intr-o zi, navigand pe mare ,o furtuna puternica ii distrugе nava si el naufragiaza pe o insula.Imediat ce realizeaza ce s-a intamplat incepe sa caute adăpost, mancare si materiale pentru a face o plută cu care sa plece,dar observa ceva bizar. Ascuns în tufisuri vede niste creaturi stranii,acesta este momentul in care el realizeaza ca a ajuns pe insula unde se afla comoara.Acum John se pregateste sa caute comoara si apoi sa scape de pe aceasta insula.



Prezentare joc:

Campanii pentru un singur jucător în care acesta trebuie să se strecoare pe lângă dusmani (sa ii evite), să se ferească de capcane și să ajungă la ieșire după ce și-a indeplinit scopul (a obținut item-urile necesare pentru a progresă și comoara însăși pentru terminarea jocului). Player ul fiind un pirat destul de lacom acesta nu va putea colecta cheile pentru fiecare nivel decât dacă a colectat toți banii de aur de pe harta.

Reguli joc:

Jucatorul are ca scop gasirea comorii ,pentru aceasta el trebuie sa evite dusmanii si capcanele si sa obtina item -urile necesare pentru a-si indeplini scopul.

Daca jucatorul intra in contact direct cu inamicii sau pica in capcanele amplasate pe harta ,acesta va fi ranit .Daca va fi ranit de prea multe ori acesta va muri ,astfel este infrant.Daca reuseste sa obtina item-urile necesare pentru terminarea nivelului sau chiar comoara(pentru terminarea jocului),el castiga.

Personajele jocului:

- **John** este protagonistul și jucătorul-personaj. El este descris ca fiind tipul hotarăt, destul de lacom ,persoana care nu renunță niciodată. Pasionat de aventura,mister,comori pierdute,el este genul de persoana care și-ar risca viața pentru a își indeplini telul.

Vestimentația sa fiind una clasica de pirat.



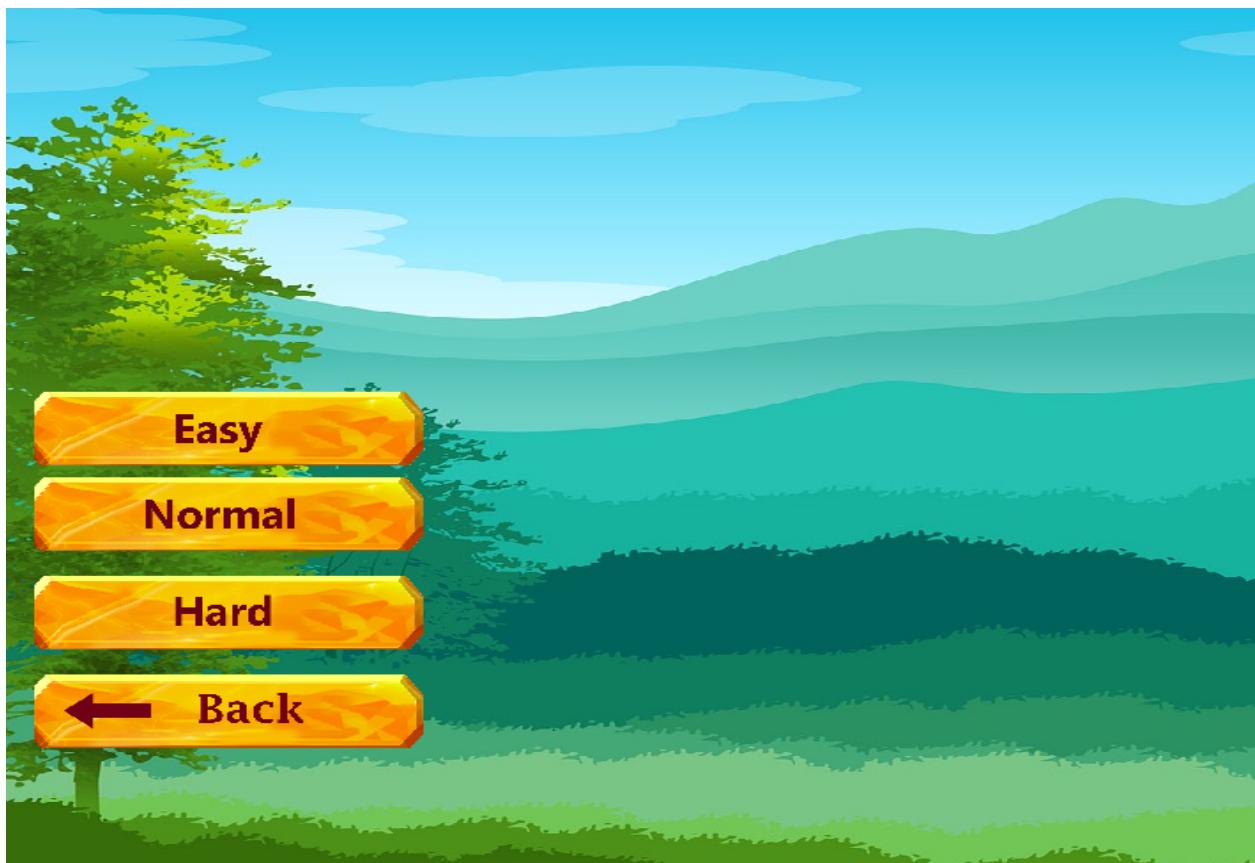
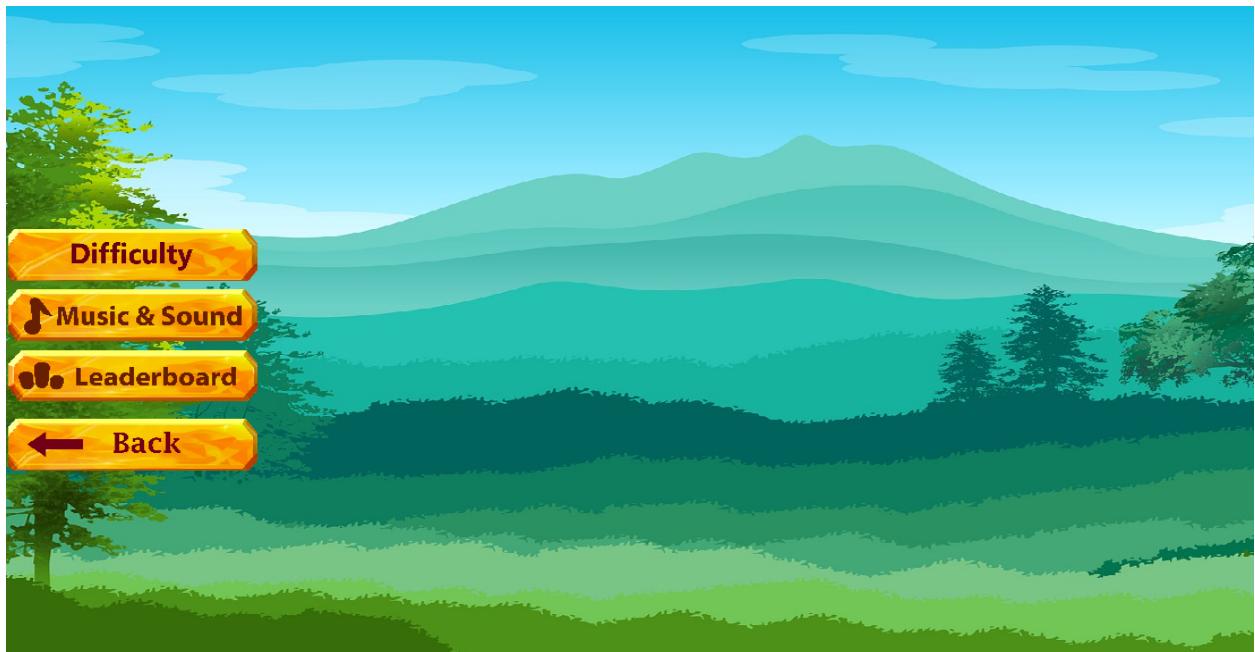
- **Creaturile(inamicii)** sunt cele care protează comoara de oamenii ce doresc să o fură,aceste creaturi diferă între ele prin aspect și abilitățile lor .Toate creaturile sunt imune la otrava.



Descriere meniu

Meniul este format din
"New Game"- permite inceperea unui joc nou ;
"Load Game"- permite continuarea unui meci inceput anterior;
"Options"-aici se poate alege dificultatea,activarea/dezactivarea sunetului si se poate vedea si tabela de scoruri ;
"Quit"-pentru parasirea programului.







In timpul jocului,jucatorul poate pune pauza la joc si sa decida daca sa se intoarca in meniu-ul principal,sa reinceapa nivelul, sa porneasca sau sa opreasca muzica sau sa se reintoarca in joc.

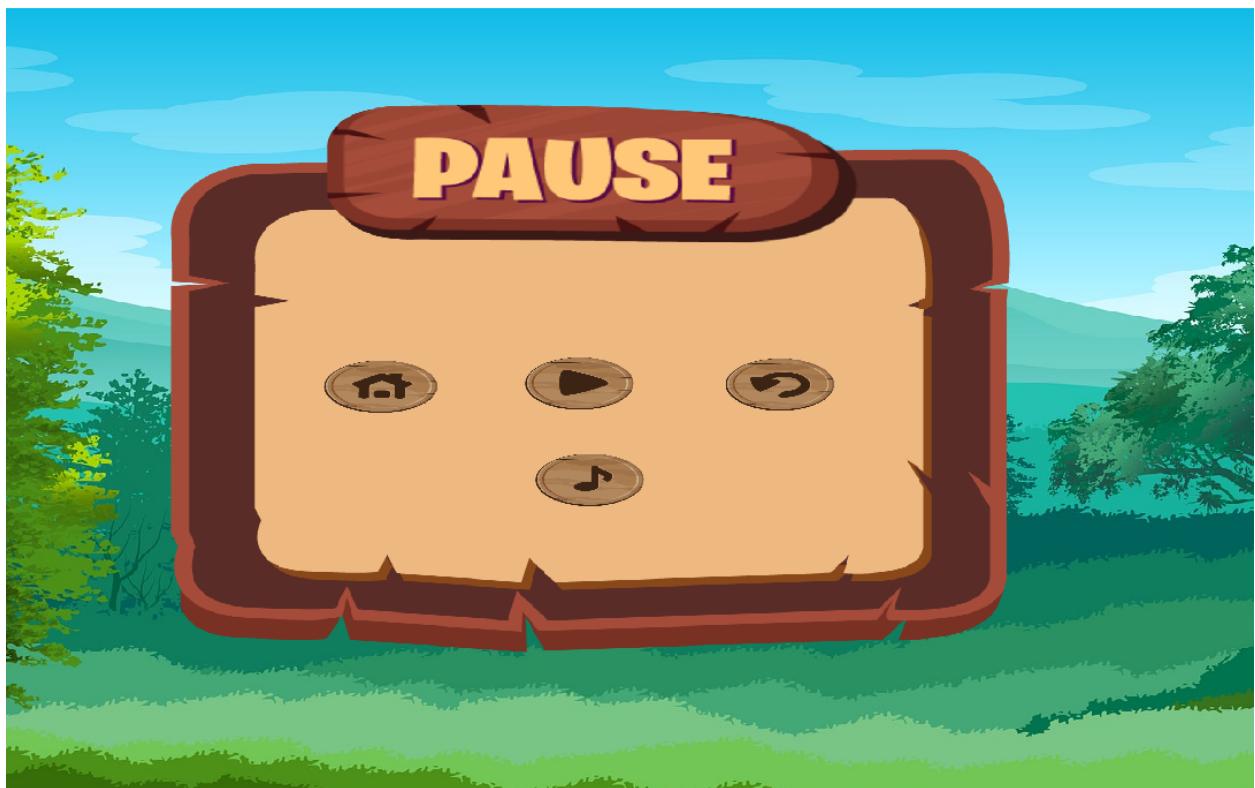
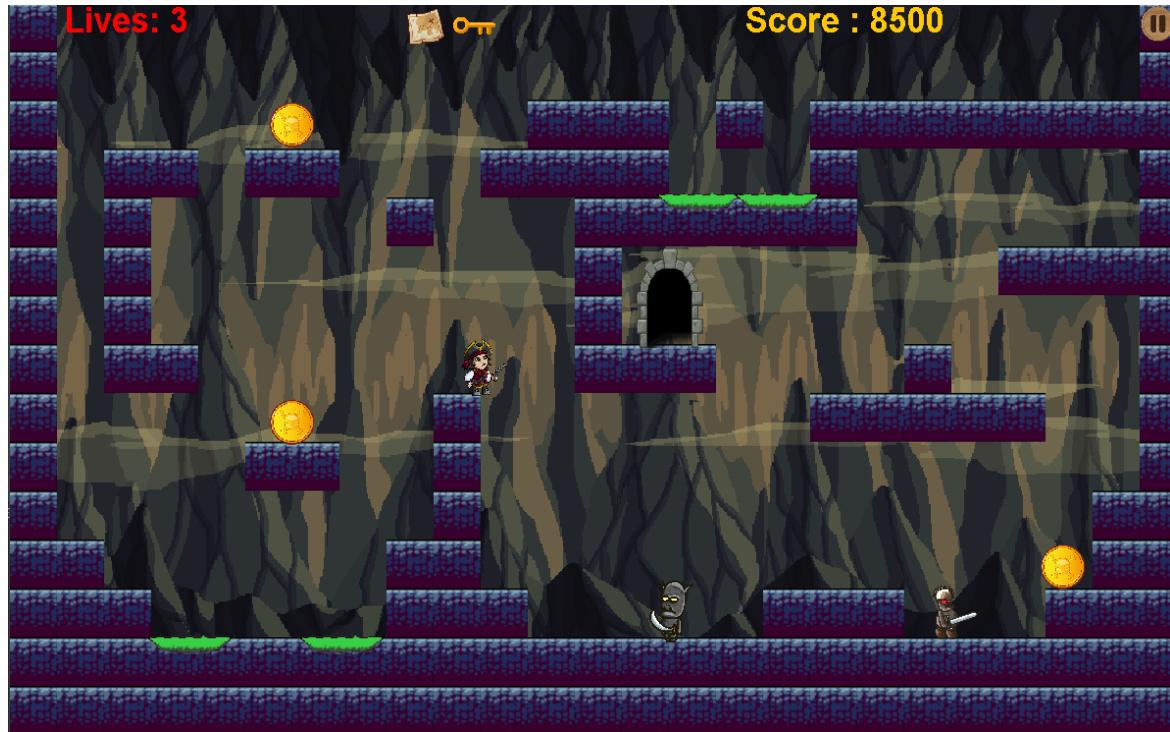


Tabla de joc /Screenshot-uri pe parcursul jocului





- Componente active : sol(block),piatra(block), ,jucatorul nu va putea trece prin acestea; alte elemente active sunt obstacolele(apa otravita) ce il va rani.
-
- Componente pasive: Banutii de aur,in momentul in care jucatorul intra in contact si apasa tasta “SPACE” vor fi colectati si vor dispara de pe harta , player-ul primind un anumit numar de puncte.Alte componente pasive sunt:cheile pentru fiecare nivel(harta,cheia comorii,comoara).
- Harta jocului este reprezentata printr-o matrice ,in functie de fiecare valoare din matrice se vor desene anumite imagini.Anumite elemente ale matricii vor reprezenta capcane,iar la anumite coordonate vor fi plasati si inamicii.Pozitille acestora vor fi diferite de la nivel la nivel(de la harta la harta).De asemenea ,in spatele acestor elemente ale jocului se afla background-ul reprezentat de mai multe imagini suprapuse pentru a oferi un peisaj complex.

Descriere nivele:

Nivelul 1-Inceputul(Explorare/Acomodare)

In acest nivel jucatorul se afla abia venit pe insula,locul este reprezentat de peisajul insulei si intrarea usor-usor in jungla.

Scopul acestui nivel este de a fi un tutorial pentru jucator ,fiind un nivel relativ usor(totusi,in meniu se poate mari dificultatea tuturor nivelurilor,inclusiv acesta)

Jucatorul se va familiariza cu felul in care se deplaseaza si interactioneaza cu diferite lucruri(colectare banuti, evitarea unor capcane...).Nivelul se termina cand jucatorul reuseste sa colecteza toti banutii , harta comorii si apoi ajunge la poarta (intrarea in jungla si deblocarea nivelului 2).

Nivelul 2-Jungla(Gasirea cheii comorii)

Peisajul acestui nivel este o jungla deasa si intunecata.

Acum jucatorul cauta indicii despre comoara,afland ca in jungla se afla cheia comorii..La finalul nivelului acesta va obtine cheia comorii si se va indrepta catre ultimul nivel.

Nivelul 3-Pestera(Obtinerea comorii)

Acest nivel contine aspectele si trasaturile unei pesteri.

Player-ul fiind la ultimul nivel doreste sa obtina comoara.La obtinerea acesteia jocul va fi finalizat.

Mecanica jocului

- **CONTROL:**

Pentru a controla miscarea personajului sunt folosite sagetile tastaturii "Left Arrow"-mers la stanga , "Right Arrow"-mers la dreapta , "Up Arrow"-saritura, Colectarea item-urile se face apasand tasta "SPACE".Pentru acces la meniu se va folosi mouse-ul.

- **SCOR:**

La inceputul jocului player-ul incepe cu scorul 0,ulterior acestea va creste atunci cand va colecta item-urile de pe harta.Fiecare obiect va oferi un anumit numar de puncte.La un scor divizibil cu 5000,player-ul va primi o viata bonus.

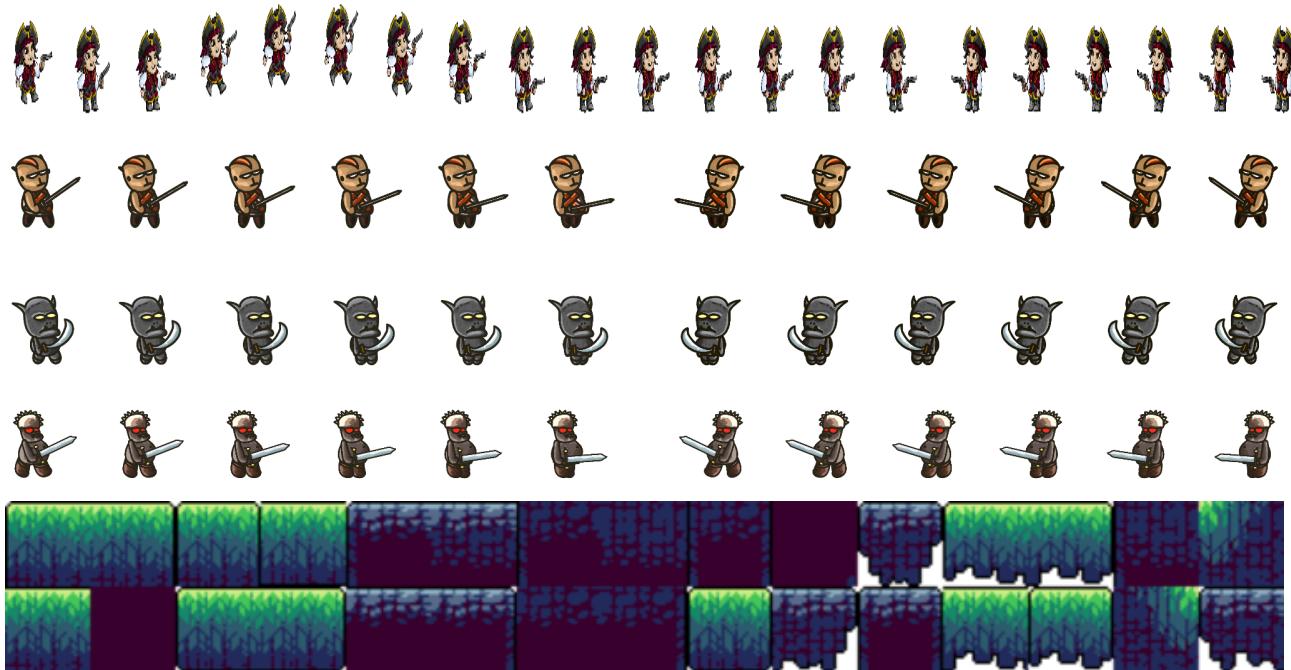
- **VIATA:.**

La inceputul jocului player-ul incepe cu trei vieti ,acestea pot creste sau descreste.Daca jucatorul intra in contact cu un inamic sau calca intr-o capcana, atunci i se va scadea o viata.Cand numarul de vieti ajunge la 0 jocul se va termina si o sa apara mesajul "YOU LOST !" si scorul obtinut.

- **FINALIZARE OBIECTIV:**

In momentul in care jucatorul atinge scopul nivelului respectiv, acesta va fi transportat la urmatorul nivel.In momentul in care termina si ultimul nivel se va afisa pe ecran mesajul "YOU WON !" si scorul final obtinut.

Game sprite



Difficulty

Difficulty



Back



Back

Easy

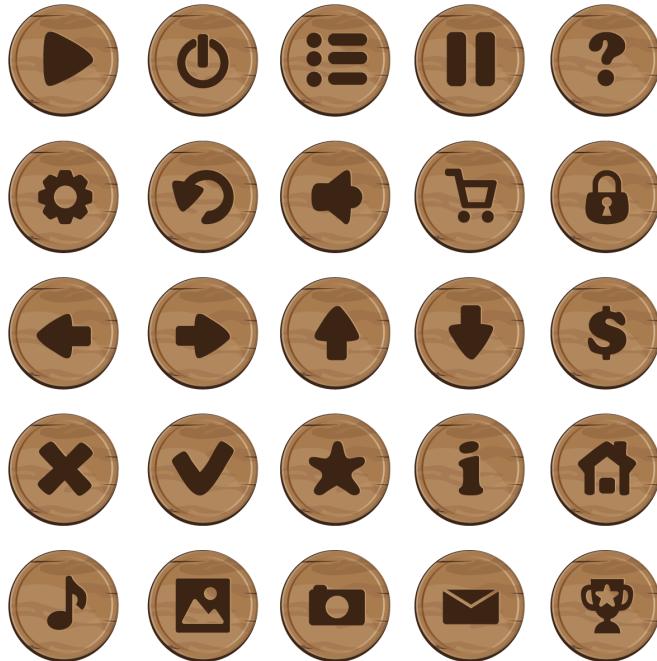
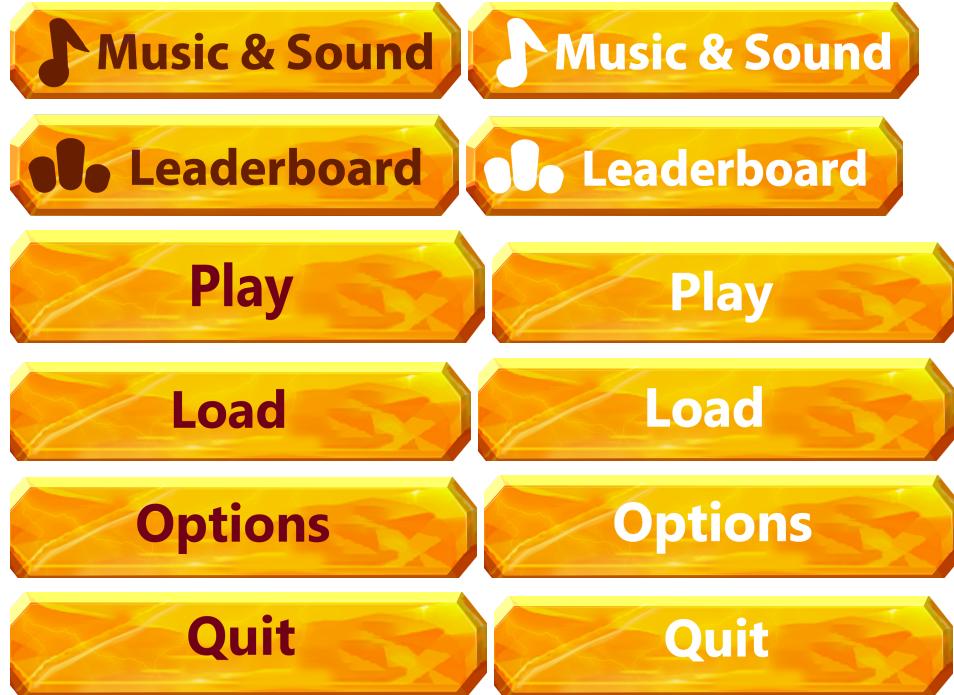
Easy

Hard

Hard

Normal

Normal

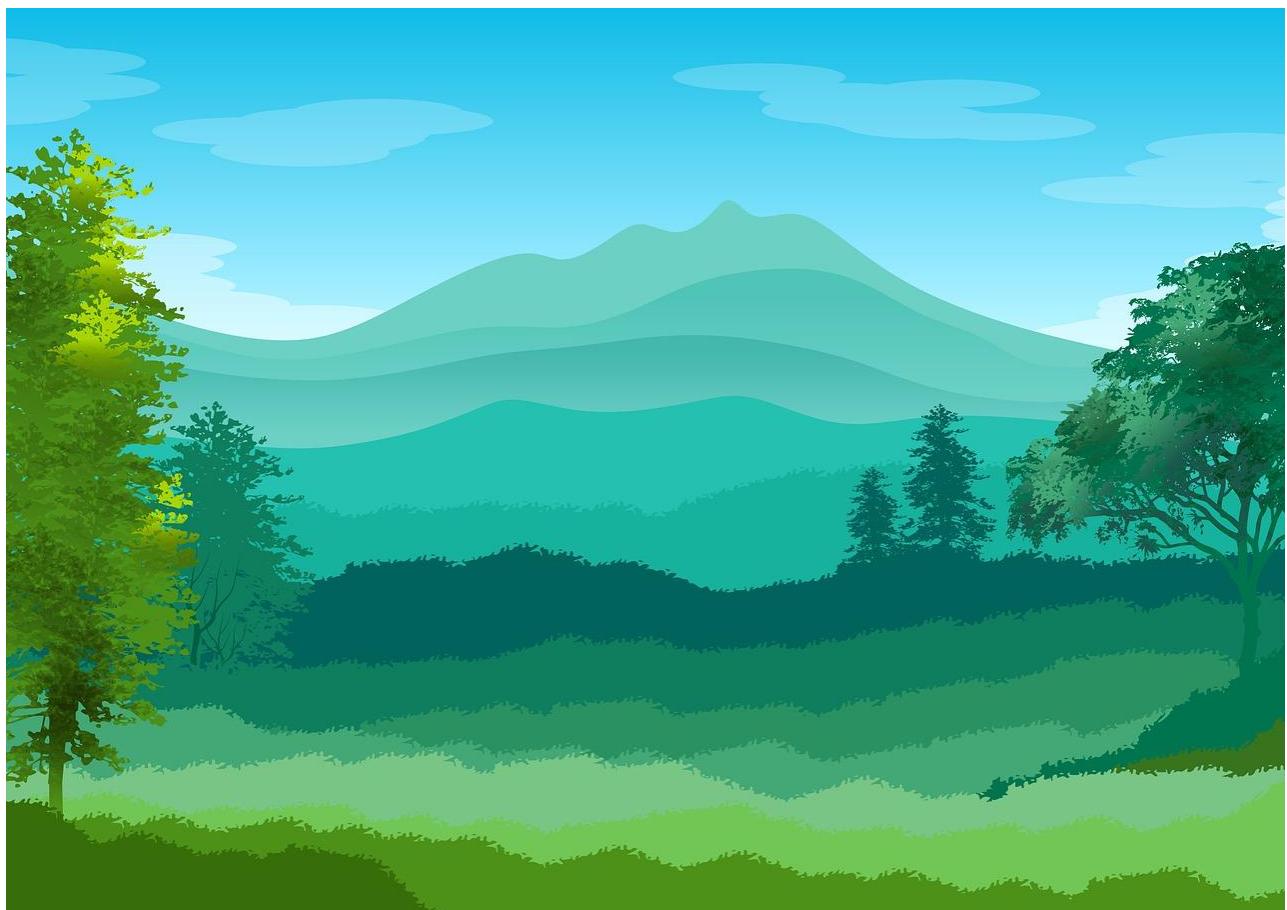
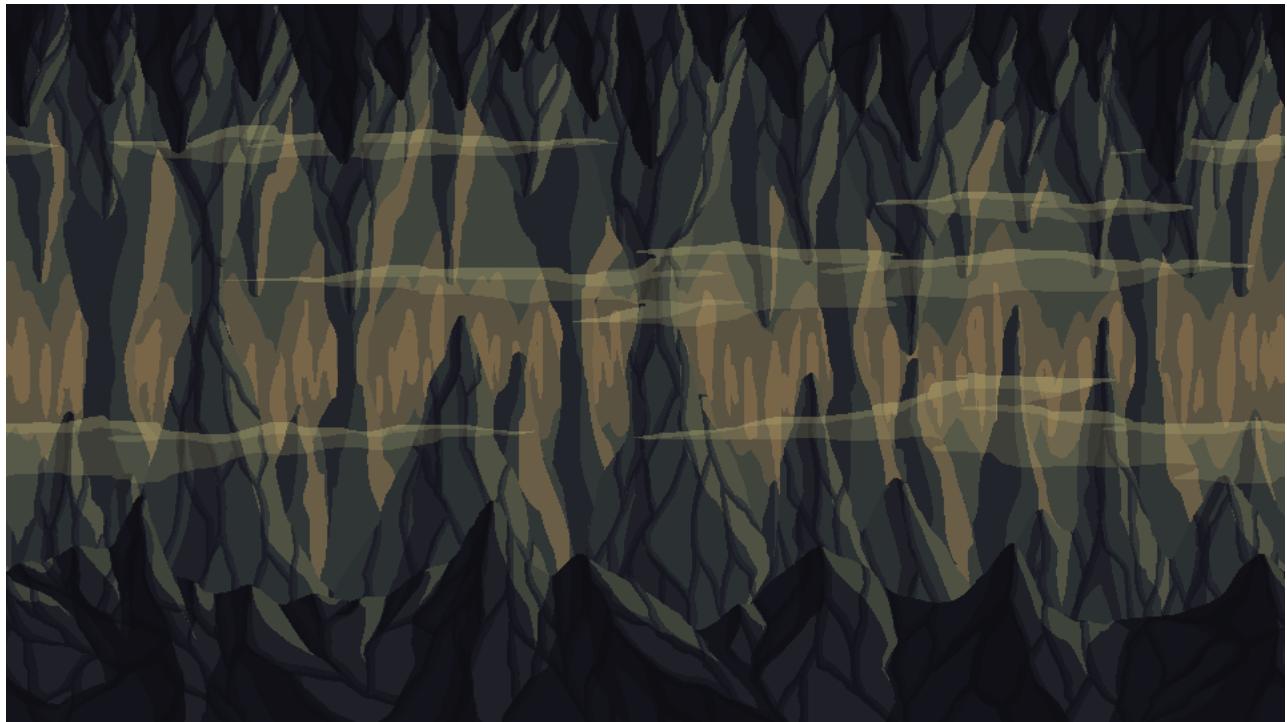


PAUSE

★ Leaderboard ★

4	
5	





Implementare joc(cod):

Items Package

Items.Item (Implementează noțiunea abstractă de entitate)
Items.Character (Definește noțiunea abstractă de caracter activ în joc)
Items.John (Implementează personajul principal al jocului)
Items.Enemy (Implementează inamicul 1)
Items.Enemy2 (Implementează inamicul 2)
Items.Enemy3 (Implementează inamicul 3)
Items.ItemsManager (Manager de entități prezente în joc)
Items_Statics.StaticEntity(Defineste noțiunea abstractă de entitate statică)
Items_Statics.CoinItem (Definește noțiunea abstractă de ban de au.)
Items_Statics.JungleDoor (Definește noțiunea abstractă de usa)
Items_Statics.BadWater (Definește noțiunea abstractă de apă otravita.)
Items_Statics.MapKey(Implementează noțiunea de cheie pentru trecerea la nivelul 2.)
Items_Statics.TreasureKey(Implementează noțiunea de cheie pentru trecerea la nivelul 3.)
Items_Statics.Treasure(Implementează noțiunea de cheie pentru terminarea ultimului nivel.)

Clasa abstractă **Item** este clasă de bază pentru clasa **Character** și **StaticEntity**, astfel acestea mostenesc atributele și metodele acestor clase.

Clasa **John** extinde clasa **Character** definind astfel personajul principal, aceasta clasa poate să arunce și o excepție de tipul **ZeroException**.

Clasa **ItemsManager** defineste managerul de entitati. Aceasta conține obiecte de tip personaj principal dar și obiecte statice, lucrând cu ele.

Metode semnificative:

Items.Item

Metode Public:

Item (RefElem refLink, float x, float y,int width, int height) Constructor de initializare al clasei
Item.void hurt(int damage) Metodă ce scade din viață atunci când există damage.

boolean checkEntityCollisions(float xOffset, float yOffset) Metoda verifică coliziunile dintre player și entitatile din joc.

Metode Protected:

boolean checkCollisions () Metoda verifică coliziunile pentru a împiedica suprapunerea cu dalele solide.

boolean isOnFloor() Metoda verifica dacă jucătorul se află pe ceva solid.

boolean isOnTop() Metoda care verifica dacă jucătorul loveste ceva cu capul.

void fall() Metoda care implementează gravitația.

Items.Character

Metode Public:

Character (RefElem refLink, float x, float y,int width, int height) Constructor de initializare al clasei Character.

void Move() Metodă ce apeleaza mai multe metode care țin de mișcarea caracterului.

void MoveX() Metoda modifică coordonata x decremetând-o atunci când personajul se deplasează la stânga,incremetând-o atunci când personajul se deplasează la dreapta.

void MoveY() Metoda modifică coordonata y decremetând-o atunci când personajul se deplasează în sus. incremetând-o atunci când creatura se deplasează în jos.

Metode Protected :

void jump() Metoda aceasta realizeaza saltul caracterului.

Items.John

Metode Public:

John(RefElem refLink, float x,float y,int width, intheight) Constructor de initializare al clasei John.

void checkObstacle() Metoda verifca coliziunea caracterului cu obstacole care pot provoca damage.

void die() Prin aceasta funcție se realizeaza diferite operatii atunci când caracterul moare.

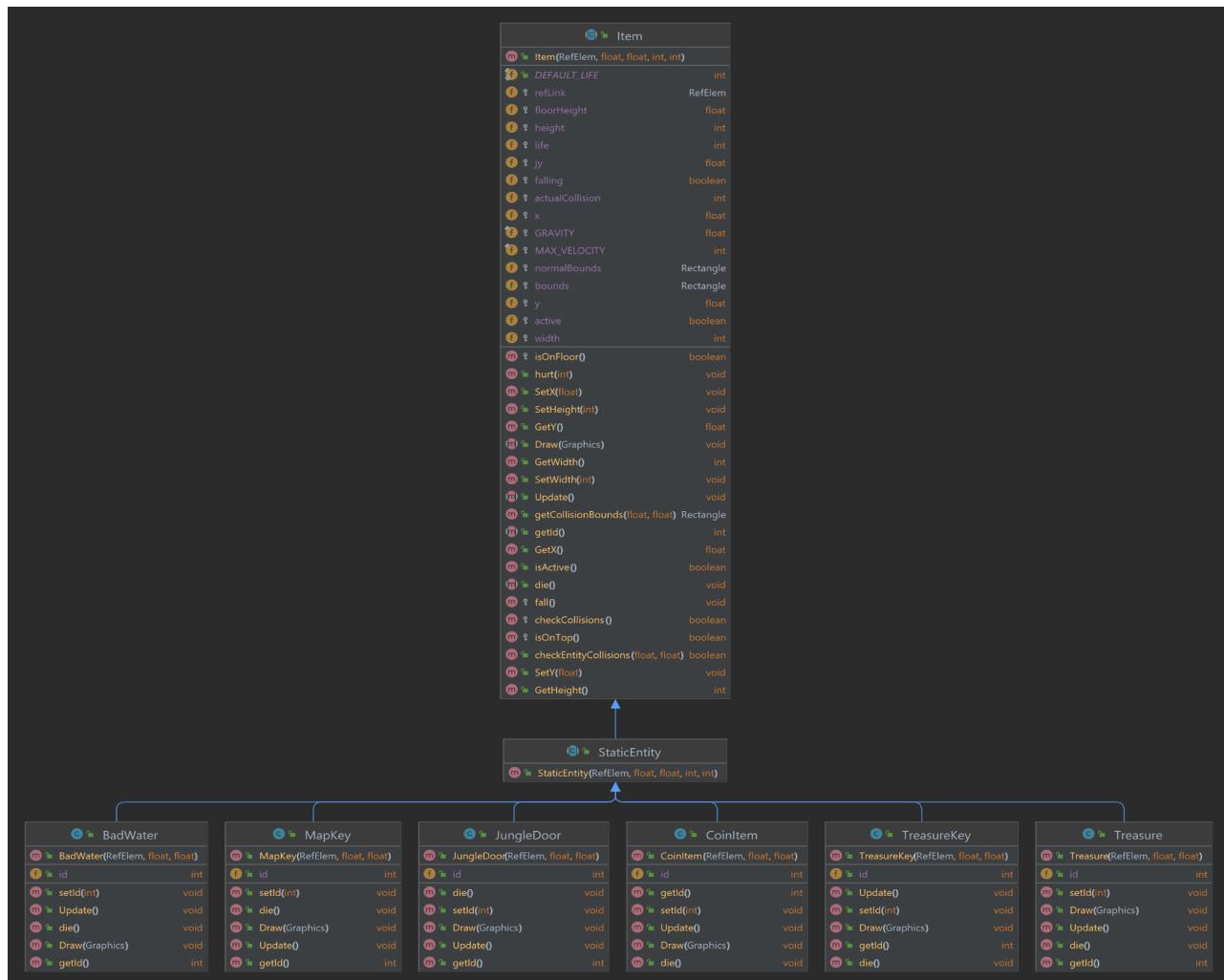
Metode Protected :

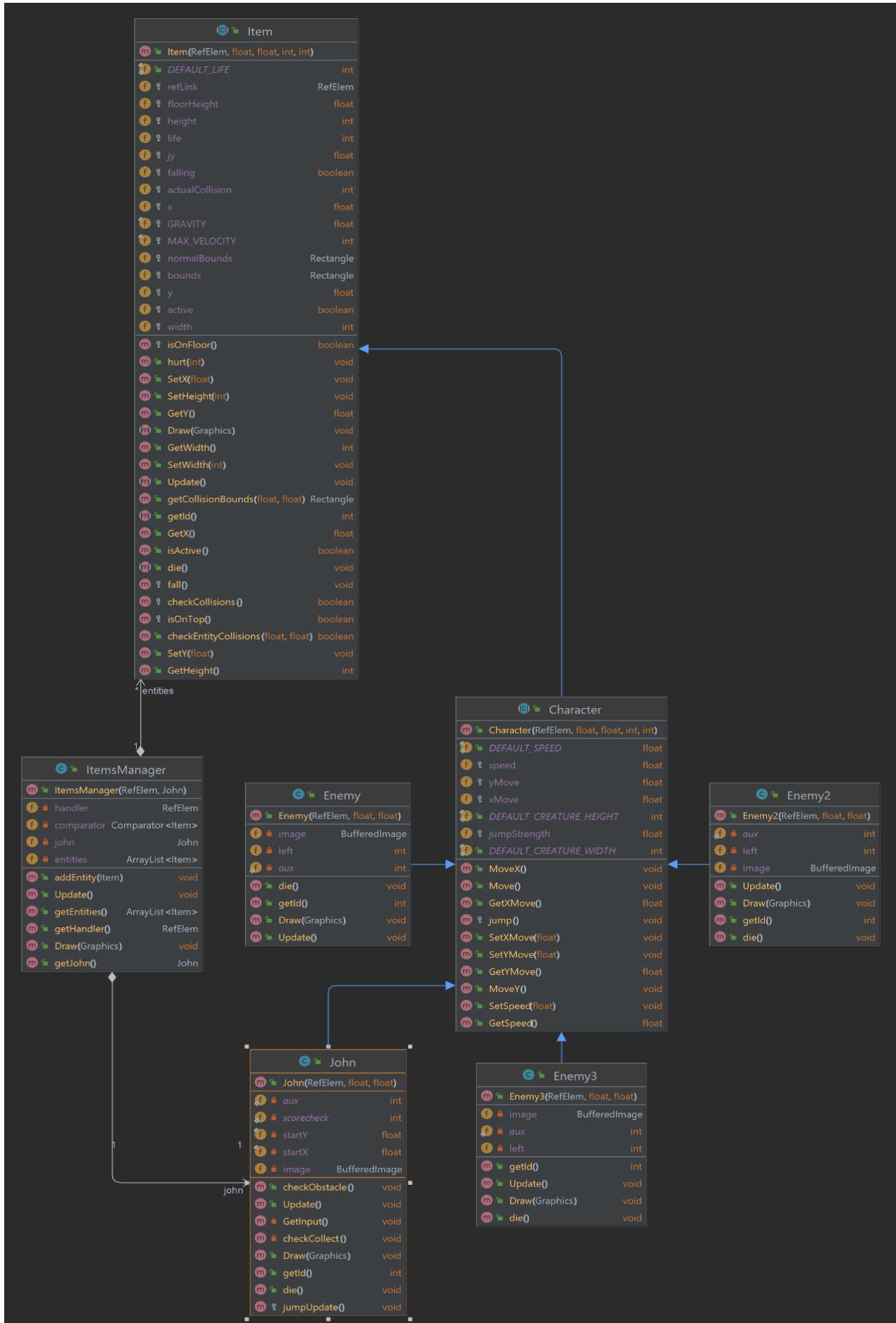
void jumpUpdate() Funcție de update a sariturii caracterului.

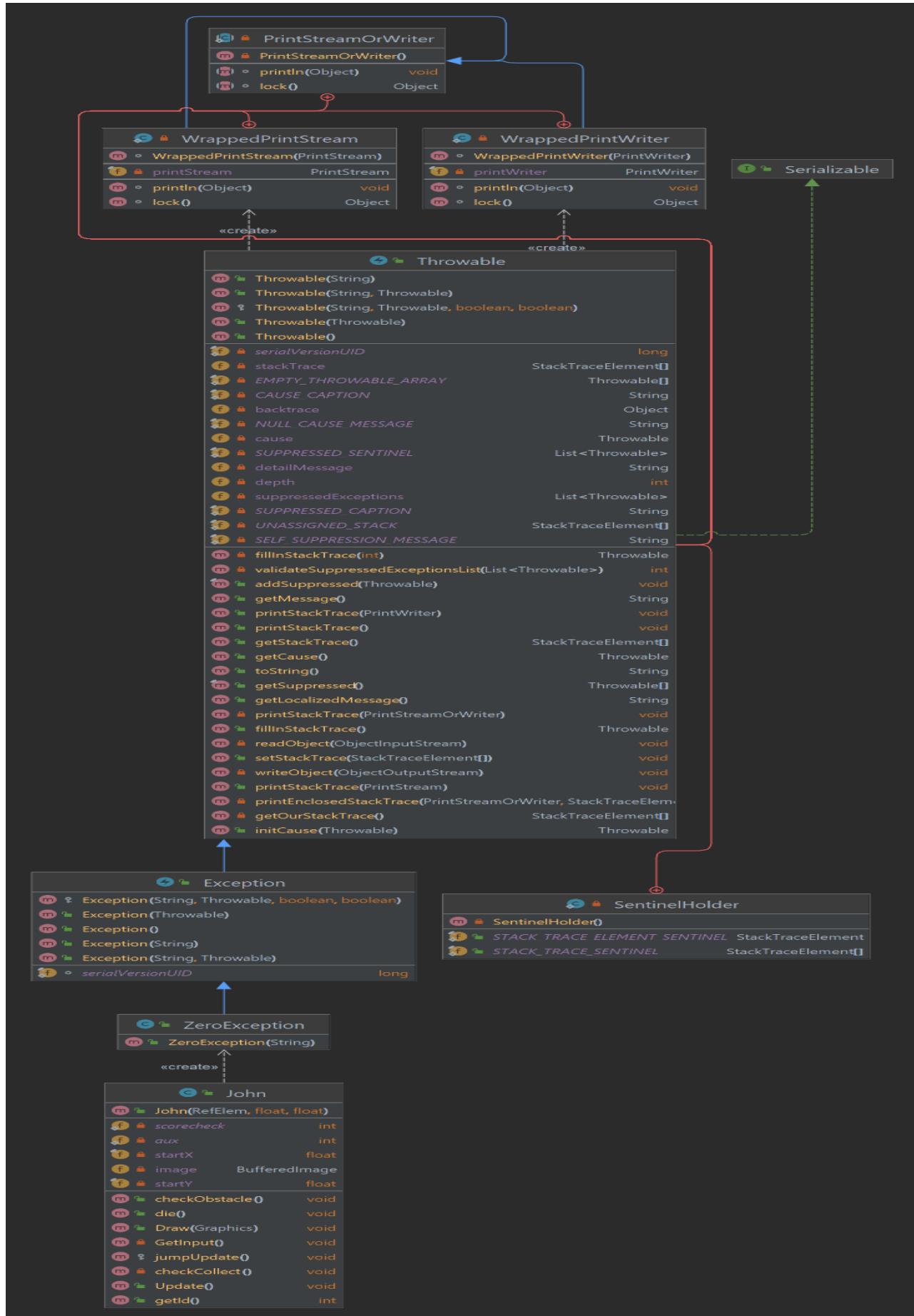
Metode Private :

void checkCollect() Metodă ce ajuta la colectarea banutilor, cheilor și trecerea la nivelul urmator.

void GetInput() Metoda verifica dacă a fost apasată o tasta din cele stabilite pentru controlul eroului.







Input Package:

Input.KeyManager (Gestionează intrarea (input-ul) de tastură)

Input.MouseManager(Gestionează intrarea (input-ul) de la mouse)

Metode Public

KeyManager () Constructorul clasei.

void update () Actualizează apăsarea tastelor.

void keyPressed (KeyEvent keyEvent) Funcție ce va fi apelată atunci când un eveniment de tastă apăsată este generat.

void keyReleased (KeyEvent keyEvent) Functie ce va fi apelată atunci când un eveniment de tastă eliberată este generat.

Clasa citește dacă a fost apăsată o tastă, stabilește ce tastă a fost acționată și setează corespunzător un flag.Dacă flagul respectiv este true înseamnă că tasta respectivă a fost apăsată și false nu a fost apăsată.

Metode Public

MouseManager () Constructorul clasei.

void setUiManager (UIManager uiManager) Setează obiectul de tip UIManager. boolean

isLeftPressed () Returnează flag-ul de click stânga.

boolean isRightPressed () Returnează flag-ul de click dreapta.

int getMouseX () Returnează poziția x a cursorului.

int getMouseY () Returnează poziția y a cursorului.

void mousePressed (MouseEvent mouseEvent) Funcție ce va fi apelată atunci când un eveniment de mouse apăsat este generat.

void mouseReleased (MouseEvent mouseEvent) Funcție ce va fi apelată atunci când un eveniment de mouse eliberat este generat.

void mouseMoved (MouseEvent mouseEvent) Funcție ce va fi apelată atunci când un eveniment de mouse este mutat.

void mouseEntered (MouseEvent mouseEvent) Funcție ce va fi apelată atunci când un eveniment de mouse introdus.

void mouseExited (MouseEvent mouseEvent) Funcție ce va fi apelată atunci când un eveniment de mouse își termină execuția.

void mouseDragged (MouseEvent mouseEvent) Funcție ce va fi apelată atunci când un eveniment de mouse este apăsat și mutat.

void mouseClicked (MouseEvent mouseEvent) Funcție ce va fi apelată atunci când un eveniment de mouse este apăsat.

Clasele **MouseManager** și **KeyManager** reprezintă metodele de interacțiune a utilizatorului cu jocul prin selectarea butoanelor, deplasarea jucătorului și colectarea bonusurilor

MouseManager		
m	MouseManager()	
f	mouseX	int
f	uiManager	UIManager
f	leftPressed	boolean
f	rightPressed	boolean
f	mouseY	int
m	mouseReleased(MouseEvent)	void
m	mouseEntered(MouseEvent)	void
m	mouseMoved(MouseEvent)	void
m	getMouseX()	int
m	mouseExited(MouseEvent)	void
m	mouseClicked(MouseEvent)	void
m	mouseDragged(MouseEvent)	void
m	getMouseY()	int
m	setUIManager(UIManager)	void
m	mousePressed(MouseEvent)	void
m	isLeftPressed()	boolean
m	isRightPressed()	boolean

KeyManager		
m	KeyManager()	
f	right	boolean
f	up	boolean
f	space	boolean
f	keys	boolean[]
f	down	boolean
f	left	boolean
m	keyPressed(KeyEvent)	void
m	keyTyped(KeyEvent)	void
m	keyReleased(KeyEvent)	void
m	Update()	void

Package GameWindow:

GameWindow.GameWindow(Implementează noțiunea de fereastră a jocului)

GameWindow		
m	GameWindow(String, int, int)	
f	wndTitle	String
f	canvas	Canvas
f	wndFrame	JFrame
f	wndWidth	int
f	wndHeight	int
m	GetCanvas()	Canvas
m	BuildGameWindow()	void
m	GetWndFrame()	JFrame
m	GetWndHeight()	int
m	GetWndWidth()	int

Package Graphics:

Graphics.Assets (Clasa încarcă fiecare element grafic necesar jocului)

Graphics.Background (Clasa ce contine backgroundul jocului)

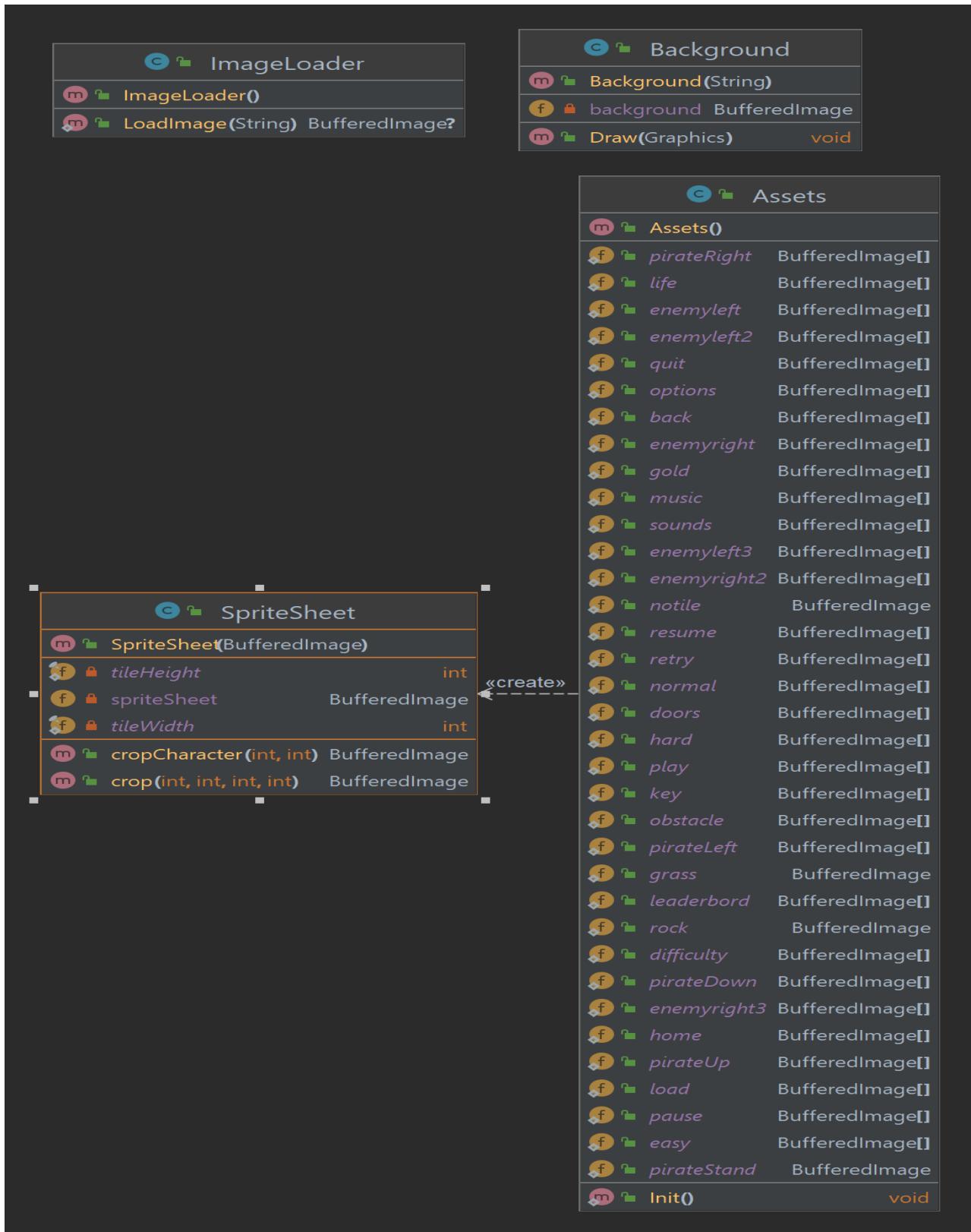
Graphics.ImageLoader (Clasa ce contine o metodă statică pentru încarcarea unei imagini în memorie)

Graphics.SpriteSheet (Clasa reține o referință către o imagine formată din dale (sprite sheet))

Metoda crop() returnează o dală de dimensiuni fixe (o subimagine) din sprite sheet de la adresa (x * lățimeDală, y * înălțimeDală).

Funcția init() inițializează referințele către elementele grafice utilizate. Această funcție poate fi rescrisă astfel încât elementele grafice încărcate/utilizate să fie parametrizate. Din acest motiv referințele nu sunt finale.

Clasa **ImageLoader** oferă posibilitatea citirii imaginilor pentru a fi randate sau folosite, clasa **SpriteSheet** încarcă imaginile și are metoda de decupare pentru dale.



Package Maps:

Maps.LevelMap(Implementeaza notiunea de harta pentru fiecare nivel)

Maps.Map1 (Implementează matricea pentru nivelul 1)

Maps.Map2 (Implementează matricea pentru nivelul 2)

Maps.Map3 (Implementează matricea pentru nivelul 3)

Maps.MapFactory (Genereaza harta in functie de optiunea aleasa) -**Modelul Factory**

Maps.Map(Implementeaza notiunea de harta a jocului(cuprinde si entitatile, nu doar dalele).

Clasa abstractă **LevelMap** este clasă de bază pentru clasa **Map1**, **Map2** și **Map3**, astfel acestea mostenesc atributele și metodele acesteia.

Clasa **MapFactory** reprezintă fabrica de harti pentru fiecare nivel în parte.

Metode semnificative:

Maps.Map

Metode Public:

Map(RefElem reflink) Constructor de initializare al clasei Map.

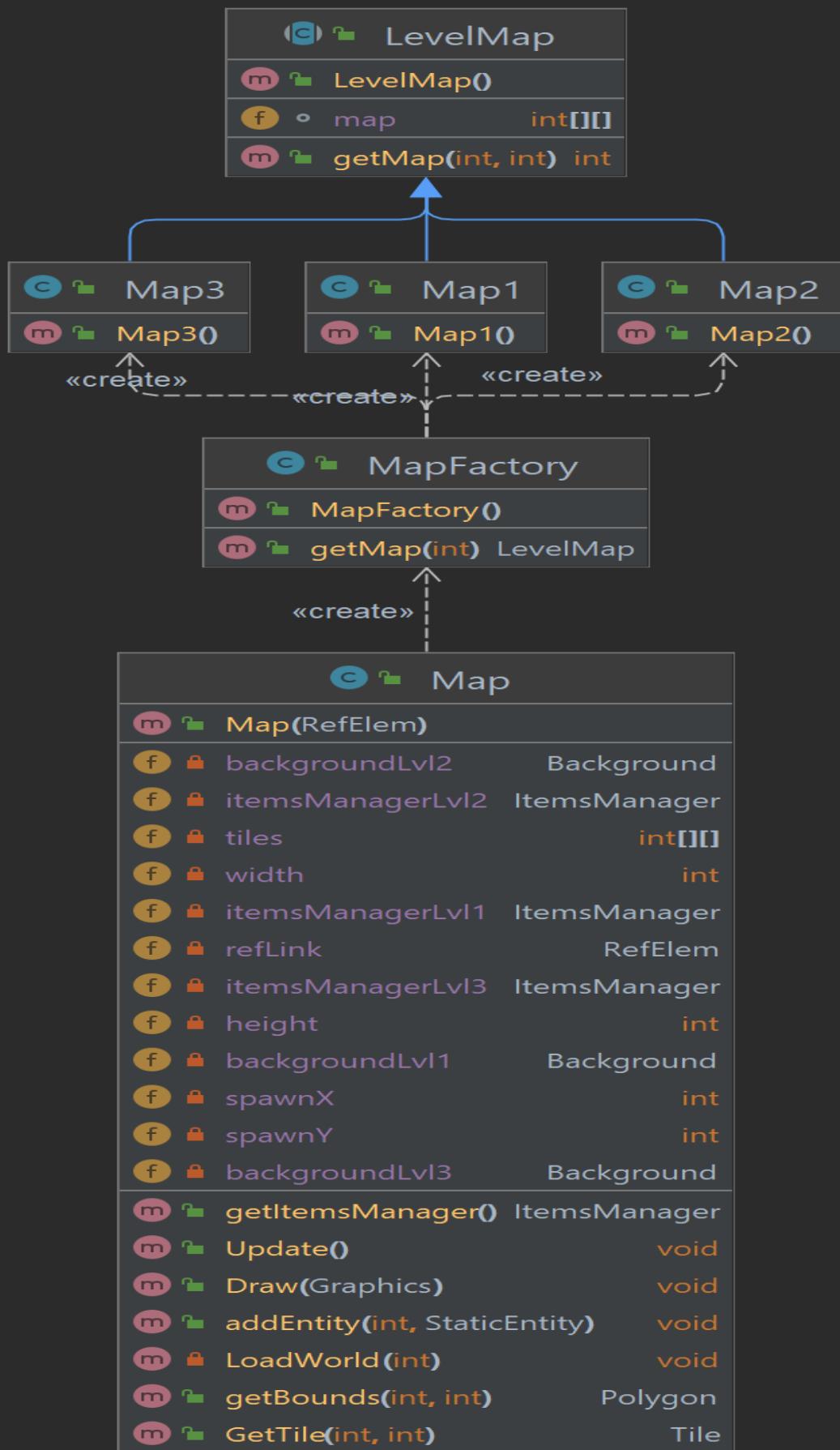
void Update() Metodă ce actualizează harta în funcție de evenimente.

void Draw(Graphics g) Metoda de desenare a hartii.

void GetTile(int x, int y) Întoarce o referință către dala aferentă codului din matrice de dale.

void LoadWord(int level) Metoda de încărcare a hartii jocului.

Polygon getBounds(int x, int y) Metoda ce returnează poligonul de coliziune a dalelor.



Package States:

States.State (Clasa abstractă ce definește noțiunea de stare în joc)

States.LeaderboardState (Implementează starea cu tabela de scoruri)

States.DifficultyState (Implementează starea de selecție a dificultății)

States.MenuState (Implementează noțiunea de meniu pentru joc)

States.OptionsState (Implementează meniul de opțiuni pentru joc)

States.PlayState (Implementează starea de play)

States.PauseState (Implementează starea de pauza)

Clasa **State** definește cele 6 stări ale jocului: **PlayState**, **MenuState**, **SettingsState**, **LeaderboardState**, **DifficultyState** și **PauseState**. Aceasta este o clasă abstractă ale cărei metode vor fi implementate de clasele ce o extind. Fiecare din acestea conțin în constructorul de initializare butoanele specifice stării actuale.

In starea **MenuState** vom găsi 4 butoane:

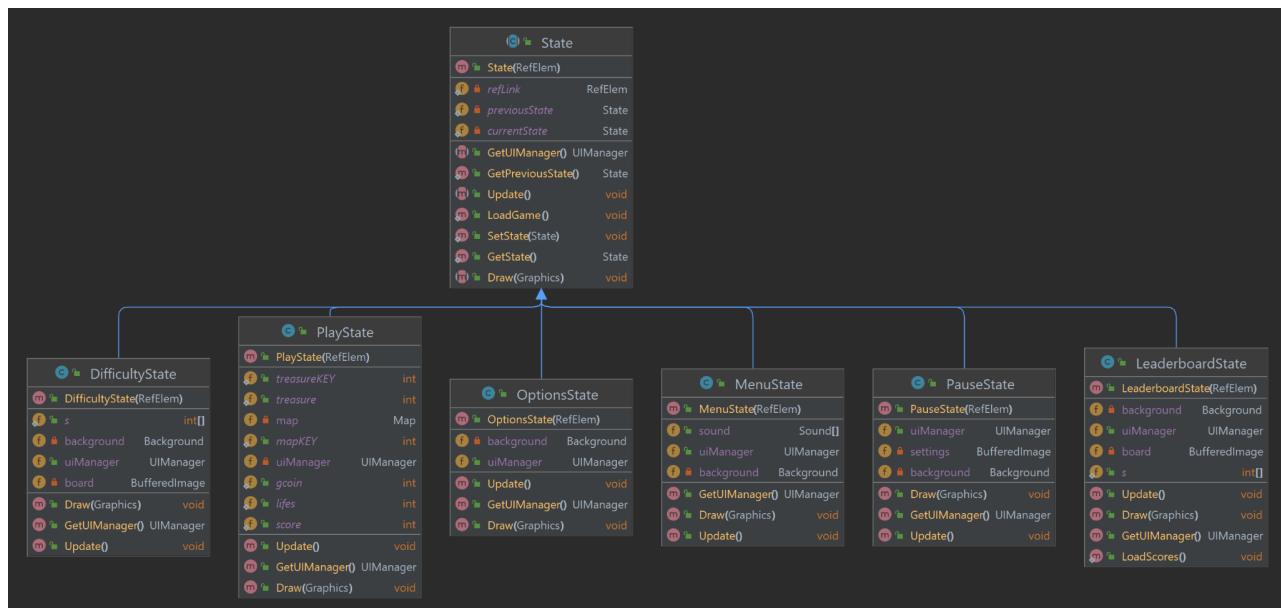
Start - pentru a începe jocul;

Load - pentru a continua de la ultimul nivel ramas(salvat in baza de date);

Options- pentru setari de dificultate,activare/oprire a sunetului si accesul la tabela cu scoruri;

Quit-pentru parasirea programului.

In **PlayState** vom găsi un buton de Pauza ,iar in starea de Pauza avem butoane pentru a ne întoarce la meniul principal,sa reincepem nivelul curent ,sa oprim/activam sunetul si sa iesim din pauza.



Package UI:

UI.ClickListener (Implementează notiunea de click în joc)
UI.UILImageButton (Implementează un obiect de tip UILImageButton pentru "User Interface")
UI.UIManager (Implementează notiunea manager de obiecte pentru "User Interface")
UI.UIObject (Implementează notiunea user interface object)
UI.Sound(Implementeaza notiunea de sunet pentru joc)
UI.Music(Implementeaza notiunea de muzica pentru joc)

Metode semnificative:

UI.UILImageButton:

Metode Public

UILImageButton (float x, float y, int width, int height, BufferedImage image, ClickListener clicker) Constructor, initializează un obiect-imagine pentru "User Interface".
void update () Actualizează starea obiectului pe "User Interface".
void draw (Graphics g) Desenează obiectul pe "User Interface".
void onClick () Implementează notiunea de click pe obiect.

UI.UIManager

Metode Public

UIManager (RefElem handler) Constructor de initializare al clasei UIManager.
void update () Actualizează obiectele de tipUIObject.
void draw (Graphics g) Desenează obiectele de tipUIObject.
void addObject (UIObject o) Adaugă un obiect.
void removeObject (UIObject o) Elimină un obiect.

UI.Sound:

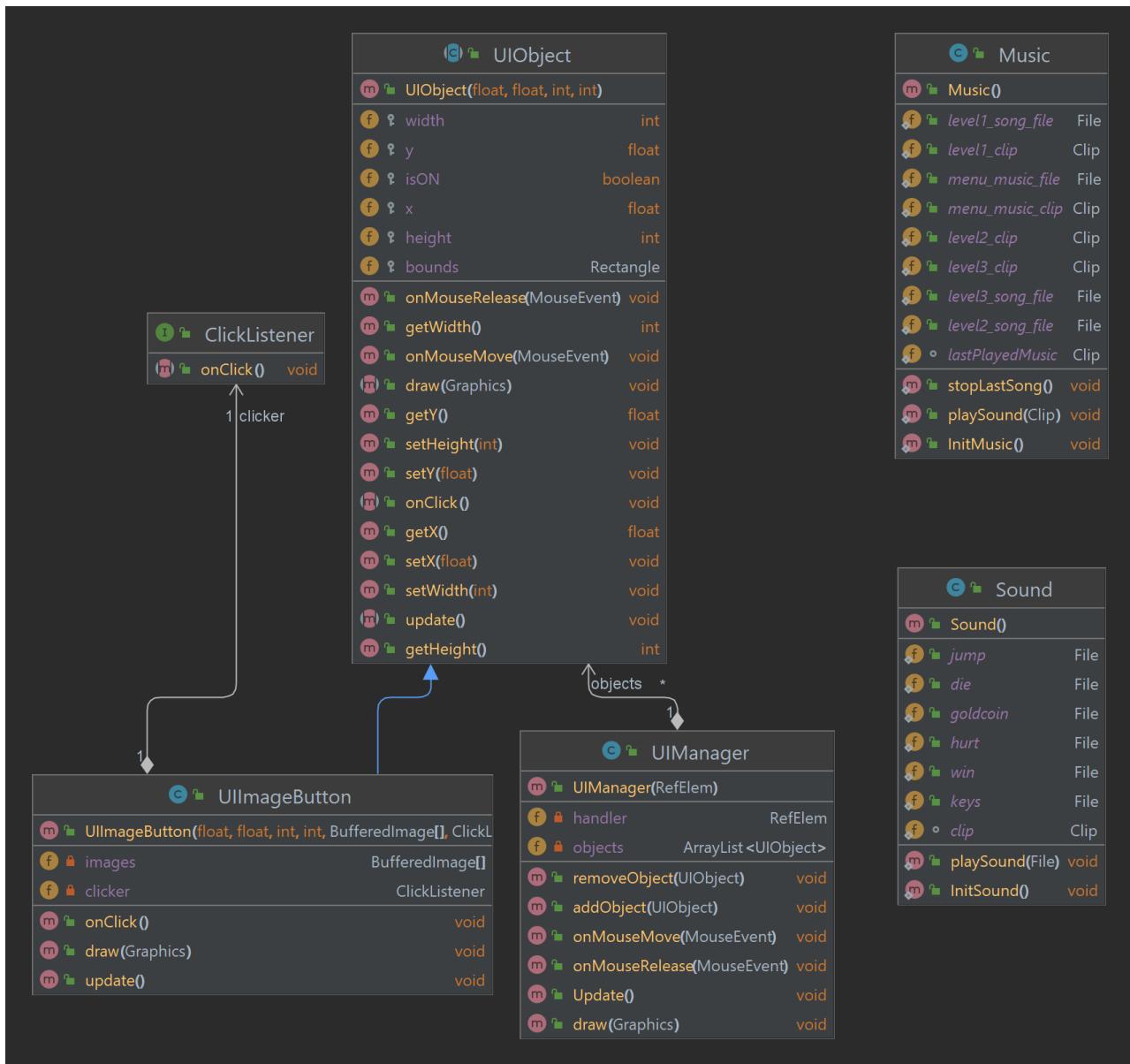
Metode statice

Public void InitSound() Initializam variabilele statice cu sunete.
void playSound(File sound) Porneste sunetul sound .

UI.Sound:

Metode statice

Public void InitSound() Initializam variabilele statice cu melodii.
void playSound(Clip sound) Porneste melodia sound..
void stopLastSong() Opreste ultima melodie.



Package Tiles:

Tiles.Tile (Clasa Tile este clasa de bază pentru clasele ce reprezintă dale, în aceasta se retine toate dalele intr-un vector și oferă posibilitatea regăsirii după un id)

TilesGrassLand(Tile de pamant acoperită cu iarbă)

Tiles.RockTile (Tile de piatra)

Tiles.Nothing (Tile gol)

Metode Public

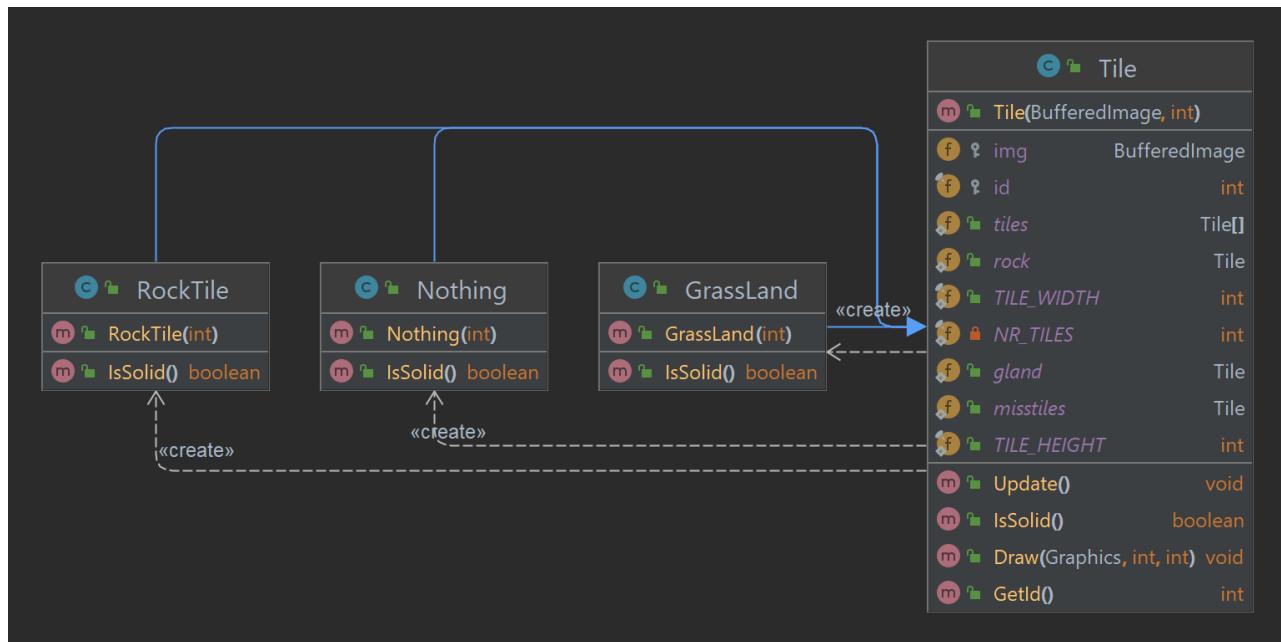
Tile (BufferedImage texture, int idd) Constructorul aferent clasei.

void update () Actualizează proprietățile dalei.

void draw (Graphics g, int x, int y) Desenează în fereastră dala.

boolean isSolid () Returnează proprietatea de dală solidă (supusă coliziunilor) sau nu.

int GetId () Returnează id-ul dalei .



Package MainGame:

MainGame.Game (Clasa principală a întregului proiect. Implementează Game - Loop (Update -> Draw))

MainGame.RefElem (Clasa ce reține o serie de referințe ale unor elemente pentru a fi ușor accesibile)

Metode Public

Game (String title, int width, int height) Constructor de inițializare al clasei Game.

State newGame () Metoda returnează o nouă referință către joc.

State continueGame () Metoda returnează o nouă referință către joc.

void run () Funcția ce va rula în thread-ul creat.

KeyManager getKeyManager () Returnează obiectul care gestionează tastatura.

MouseManager getMouseManager () Returnează obiectul care gestionează mouse-ul.

synchronized void start () Crează și startează firul separat de execuție (thread).

synchronized void stop () Opresc execuția thread-ului.

int GetWidth () Returnează lățimea ferestrei.

int GetHeight () Returnează înălțimea ferestrei.

State getGameState () Returnează referința către joc.

Metode Statice Public

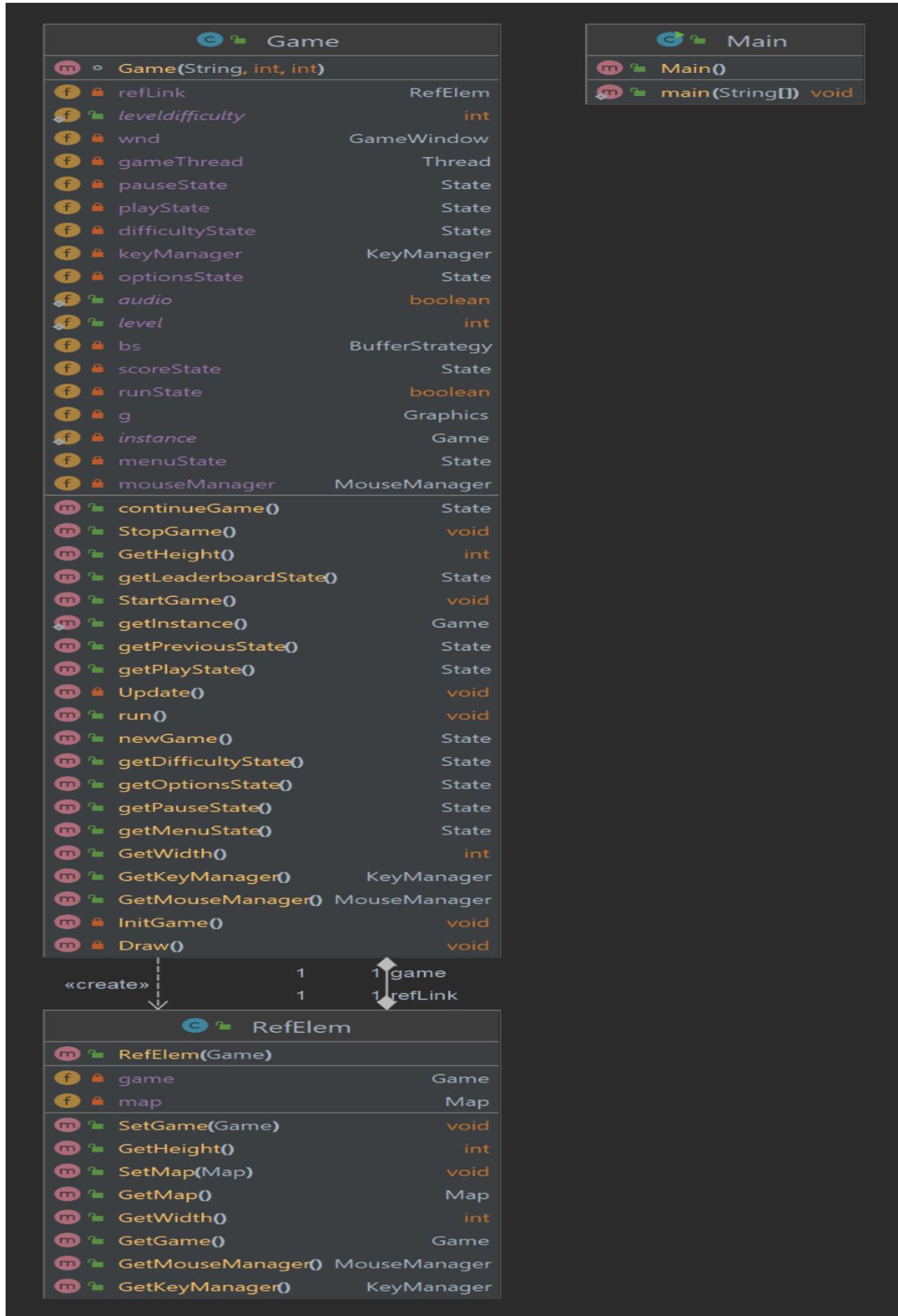
static Game getInstance () Metoda returnează un obiect de tip Game(**Modelul Singleton**).

Metode Private

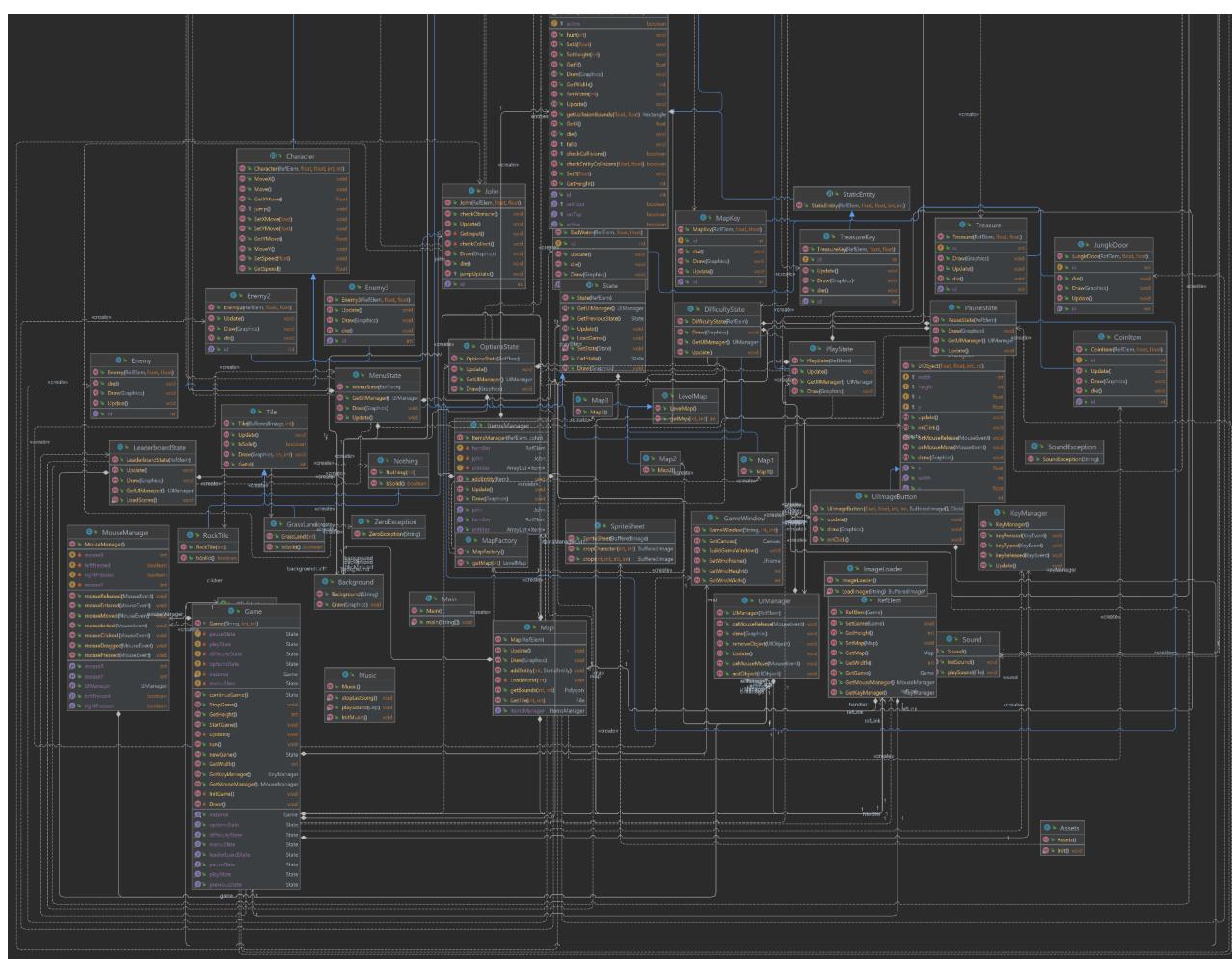
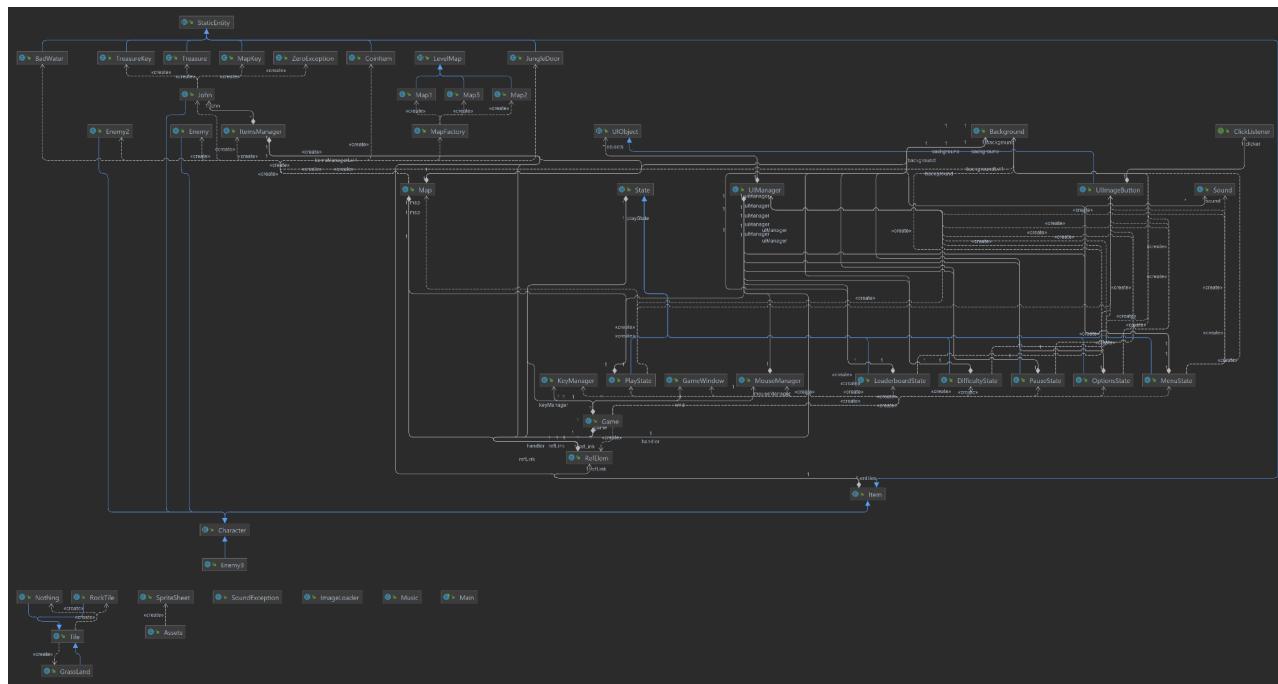
void InitGame() Metoda construiește fereastra jocului, initializeaza asset-urile, listener-ul de tastatura etc.

private void Update() Actualizeaza starea elementelor din joc

private void Draw() Deseneaza elementele grafice in fereastra corespontator starilor actualizate ale elementelor.



Diagramă UML:



Baze de date

Pentru acest joc am folosit 2 baze de date.

1. load_game contine valorile pentru : ultimul nivel la care a ajuns jucatorul, numarul de vieti, scorul si daca a obtinut cheile pentru nivelul 1 si 2.

The screenshot shows a database interface with a toolbar at the top containing 'New Database', 'Open Database', 'Write Changes', 'Revert Changes', and 'Open Proj'. Below the toolbar are tabs: 'Database Structure', 'Browse Data' (which is selected), 'Edit Pragmas', and 'Execute SQL'. A sub-toolbar below the tabs includes icons for refresh, search, filters, and export. The table name 'LOAD_GAME' is selected in the 'Table' dropdown. The table structure is displayed with columns: ID, AVED_FILE, SAVED_SCORE, SAVED_NR_LIFES, KEY_1, and KEY_2. Each column has a 'Filter' button. A single row of data is shown:

ID	AVED_FILE	SAVED_SCORE	SAVED_NR_LIFES	KEY_1	KEY_2
1	1	1	0	3	0
1	1	1	0	3	0

2. scores_game contine ultimele 5 scoruri sortate in ordine descrescatoare, aceste valori fiind folosite pentru tabela cu scoruri.

The screenshot shows a database interface with a toolbar at the top containing 'New Database', 'Open Database', 'Write Changes', and 'Revert Changes'. Below the toolbar are tabs: 'Database Structure', 'Browse Data' (selected), 'Edit Pragmas', and 'Execute SQL'. A sub-toolbar below the tabs includes icons for refresh, search, filters, and export. The table name 'SCORES' is selected in the 'Table' dropdown. The table structure is displayed with columns: ID and score. Each column has a 'Filter' button. A table of 5 rows is shown:

ID	score
1	15000
2	15000
3	1000
4	500
5	0

Bibliografie

Pentru a face Sprite Sheet-urile:

<https://www.codeandweb.com>

Imagini pentru joc:

<https://aamatniekss.itch.io>

<https://craftpix.net>